MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI STUDIJ RAČUNARSTVA

MATIJA RISEK

INTEGRACIJA ASP.MVC APLIKACIJE S OFFICE 365 PLATFORMOM

ZAVRŠNI RAD

ČAKOVEC, 2015.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI STUDIJ RAČUNARSTVA

MATIJA RISEK

INTEGRACIJA ASP.MVC APLIKACIJE S OFFICE 365 PLATFORMOM

AN INTEGRATION OF ASP.MVC APPLICATION WITH OFFICE 365 PLATFORM

ZAVRŠNI RAD

MENTOR:

mr. sc. Bruno Trstenjak

ČAKOVEC, 2015.

Sadržaj Završnog rada:

SAŽETAK

Cilj ovog završnog rada je izraditi modernu ASP.MVC 5 aplikaciju integriranjem dostupnih servisa iz Office 365 platforme.Upotrebom navedenih, ali i drugih aktualnih tehnologija, nastojalo se automatizirati poslovni proces jedne tvrtke čija zadaća je bila provjera unosa dnevnica u sustav. U tu svrhu korišten je Microsoft Exchange 2013 servis za slanje elektroničke pošte i Microsoft Azure platforma sa ostalim servisima poput Web App-a, Scheduler-a ili SQL Database koji služe za razvoj aplikacije, njeno pokretanje i upravljanje bazom podataka. Mobilna i web aplikacija omogućuje registraciju i logiranje korisnika u sustav, unos dnevnice u bazu, te provjeru unosa koja rezultira slanjem e-maila na adresu svakog korisnika koji dnevnicu nije predao na vrijeme. U prvom dijelu rada opisan je framework ASP.MVC-a i glavna svojstva njegove arhitekture. Drugi dio rada sastoji se od opisa integrirane Office 365 platforme i njezinog Exchange servisa, te su objašnjeni ostali servisi iz Azure platforme. Treći dio rada zadrži opis klasičnih tehnologija koje su korištene pri razvoju aplikacije, te su prikazani njezini moduli. Završni dio rada opisuje proces testiranja aplikacije te sigurnosne elemente koji su ugrađeni u aplikaciju.

Ključne riječi: *ASP.MVC aplikacija, Office 365 platforma, Microsoft Exchange 2013, Microsoft Azure, Web App, Scheduler, SQL Database*

Slika 1. *Microsoft .NET MVC logo*

(http://webapplayers.com/inspinia_admin-v2.2/img/mvc_logo.png)

1. UVOD

U zadnje vrijeme često se susrećemo sa pojmovima poput modernog weba ili izvršavanja u oblaku (engl. *cloud computing*). Riječ je o novim izrazima koji su tu zahvaljujući velikim promjenama koje su nastupile posljednjih godina. Danas je uz dominaciju mobilnih uređaja i tableta za adekvatan razvoj web aplikacija potrebno razmišljati na posve drugačiji način. Pojam moderni web zapravo možda najbolje opisuje situaciju u kojoj stolna računala više ne igraju glavnu ulogu na tržištu [1]. S obzirom na te promjene, prilikom razvoja suvremene aplikacije nužno je obratiti pažnju na usmjerenost prema izvršavanju u oblaku. Većina popularnih aplikacija zato danas sadržava moderne standarde poput HTML5 i CSS3, a njihovo upravljanje, pa čak i razvoj odvija se putem Clouda, bilo da se radi o web ili mobilnim aplikacijama.

Cilj ovog Završnog rada je izraditi modernu ASP.MVC aplikaciju koja će uz pomoć integracije sa servisima Office 365 platforme korisniku omogućiti upravo takvu suvremenost. Iz perspektive razvojnog programera, uporabom Microsoft Azure platforme i njezinih servisa na primjeru sustava za unos dnevnica prikazat će se olakšani razvoj, održavanje i testiranje same aplikacije, dok će se integracija ASP.MVC aplikacije i Office 365 platforme upotrijebiti za funkcionalnost provjere i obavještavanja korisnika putem e-maila ukoliko njegova dnevnica nije unesena u predviđenom roku.

2. ZAHVALA

Ovim putem želim zahvaliti mentoru mr. sc. Bruni Trstenjaku na podršci, strpljenju i suradnji za vrijeme mog pohađanja preddiplomskog stručnog studija računarstva na Međimurskom Veleučilištu u Čakovcu, kao i svim ostalim profesorima. Zahvaljujem i šefu Velimiru Sanjkoviću na povjerenju, uloženom trudu i vremenu, ali i ostalim kolegama iz tvrtke. Također bih želio uputiti posebnu zahvalu mojim roditeljima koji su mi bili i ostali najveći uzori u životu. I konačno, ovaj Završni rad posvećujem supruzi Maji i sinu Luki bez kojih ne bih uspio.

3. CILJ I DOPRINOS ZAVRŠNOG RADA

Cilj ovog Završnog rada je izrada web i mobilne ASP.MVC aplikacije uz integraciju servisa iz Office 365 platforme. Gotova aplikacija pruža mogućnost unosa dnevnica u sustav od strane registriranih korisnika te automatizirani način kontrole njihovih unosa. MVC aplikacija je sposobna sama prepoznati koji od korisnika nije unio dnevnicu te ga o tome obavijestiti putem elektroničke pošte zahvaljujući integraciji sa servisom Microsoft Exchange Server 2013.

Izradom ove aplikacije razviti će se pouzdan, moderan i nadogradiv sustav koji će služiti za automatizaciju poslovnih procesa jedne tvrtke. Još jedan od ciljeva je napraviti web i mobilnu MVC aplikaciju koja će držati korak sa vremenom uz pomoć aktualnih tehnologija poput MVC 5 okvira čiji logo je prikazan na slici 2. Ovo je za mene posve novo iskustvo i prvi poslovni projekt koji radim što mi predstavlja izazov.

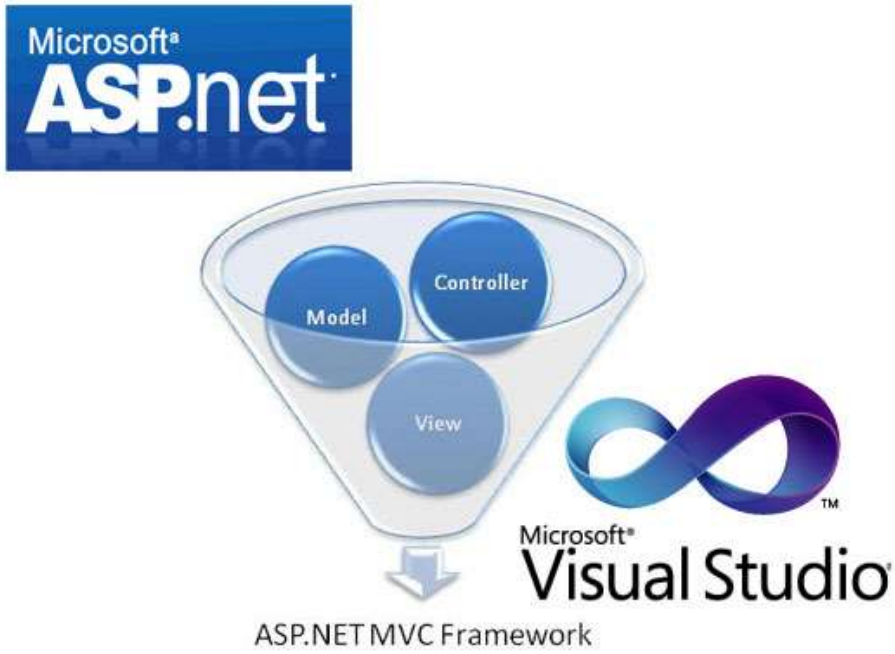
Slika 2. *Logo Microsoft .NET MVC 5 verzije*

(http://www.deccansoft.com/Documents/CourseImages/6155d0bb-524d-43c0-b07e-86d073e7b82b_newMVC.jpg)

4. MVC ARHITEKTURA

MVC ili model-prikaz-kontroler je specifična arhitektura iz područja računarstva koju kakrakteriziraju razdvojeni dijelovi web aplikacije što je i prikazano na slici 3., a svaki od tih slojeva se bavi nekom od zadaća razdijeljenih na više razina unutar same Web aplikacije. Te razine se nazivaju: model koji predstavlja podatke i logiku, prikaz koji generira vizualnu reprezentaciju i kontroler koji služi za interakciju između korisnika i sustava. Karakteristično za ASP.NET MVC arhitekturu je to što je logika aplikacije razdvojena u posebnim klasama sa zasebnim zadaćama te se nikad ne nalazi na istom mjestu gdje je i kod za pristupanje podacima. Takva vrsta predloška sa raspodjelom odgovornosti također se koristi i u programskim jezicima Java i C++, a prvi put se spominje krajem 70-ih godina kod aplikacija pisanih programskim jezikom SmallTalk od strane programera iz centra Xerox PARC [2].

Slika 3. Osnovne značajke ASP.NET MVC okvira



(http://aspnetmvceuropeanhosting.hostforlife.eu/image.axd?picture=2014%2F6%2Faspnetmvc.jpg)

4.1. Svojstva MVC arhitekture

Neka od najvažnijih svojstava MVC arhitekture su raspodjela odgovornosti ili labava sprega i mogućnost agilnog iterativnog razvoja baziranog na testiranju (engl. *test-driven development*). Testiranje je moguće izvesti bez pokretanja kontrolera zahvaljujući simulaciji objekata iz aplikacije što testiranje čini brzim i fleksibilnim. Sve komponente ASP.MVC okvira dizajnirane su na način da mogu biti zamijenjene i prilagođene, primjerice URL usmjeravanje ili serializacija akcijskih metoda. MVC okvir podržava DI (engl. *Dependancy injection*) i IOC (engl. *Inversion of control*) uz pomoć kojih je moguće ubaciti objekte izravno u razred, i referencirati se na neki drugi objekt iz vanjskog izvora odnosno konfiguracijske datoteke što olakšava testiranje. URL mapiranje odlikuje se SEO (engl. *search engine optimization*) i REST (engl. *representational state transfer*) podrškom koja omogućuje razvoj aplikacije složenim URL-ovima koje je lako pretraživati. Prestala svojstva MVC arhitekture su podržana Windows autentifikacija, URL autorizacija, uloge (engl. *roles*), razni predlošci i tako dalje [3].

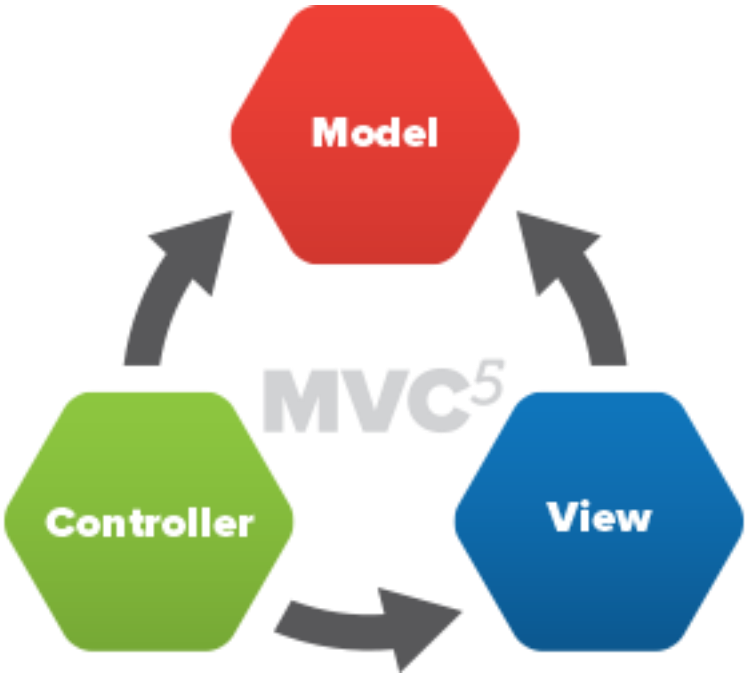
4.2. Prednosti i nedostaci

Glavna prednosti MVC okvira su olakšani razvoj, upravljanje i testiranje aplikacije zahvaljujući podjeli aplikacije na model, prikaz i kontroler. MVC framework daje punu kontrolu nad ponašanjem aplikacije, sadrži bogat mehanizam za usmjeravanje i pogodan je za veće ekipe razvojnih programera. Nedostatak MVC okvira je kompleksnost za razliku od Web forms pristupa jer se kod jedne stranice dijeli na minimalno tri datoteke. Ne postoji „Povuci i potegni“ (engl. *Drag-and-drop*) način izrade obrasca pa je takav kod potrebno pisati ručno. Alat Microsoft Visual Studio zapravo uopće ne nudi rješenje za vizualno dizajniranje MVC prikaza uporabom Razor sintakse.

4.3. Slojevi arhitekture

Kod MVC arhitekture aplikacija se sastoji od ukupno tri glavna sloja, a to su: model, prikaz i kontroler. Kao što je prikazano na slici 4. svaki od navedenih slojeva je zadužen za specifične zadatke i radi svoju zadaću neovisno od ostalih.

Slika 4. *Glavne komponente MVC okvira*



(http://www.markafarrugia.com/wp-content/uploads/2012/10/MVC.jpg)

4.3.1. Kontroleri

Mehanizam koji korisniku omogućava interakciju sa sustavom zove se kontroler, sastoji se od funkcija i akcija koje se aktiviraju kao reakcije na temelju određenog korisničkog unosa. Zato je kontroler odgovoran za interpretaciju i razmjenu poruka između prikaza i modela. Kontroler zapravo smatramo specijalnim razredom (*engl. Class*) koji služi za koordinaciju odnosa između modela i prikaza. U ASP.MVC okviru postoji konvencija da se nazivu te klase dodaje sufiks „Controller“, a na primjer klase kontrolera iz aplikacije možemo vidjeti na kodu 1.

Kod 1. Primjer klase kontrolera u MVC aplikaciji

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace WebAplikacijaMvc.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult About()
        {
            ViewBag.Message = "Ovdje se obavlja unos
dnevnice.";

            return View();
        }

        public ActionResult Contact()
        {
            ViewBag.Message = "Matija Risek";

            return View();
        }
```

4.3.2. Prikazi

Za dinamičko vizualno reprezentiranje svakog pojedinog modela zadužen je prikaz. Dakle, koristimo ga za prikaz podataka, ali i da bismo korisniku omogućili izmjenu tih podataka. Vizualna reprezentacija je zapravo interpretacija generiranog HTML koda u Web pregledniku (*engl. Web Browser*), međutim ne smijemo zaboraviti da je također moguća vizualizacija istog modela u raznolikim formatima poput HTML-a, XML-a ili npr. PDF-a. Ranije spomenuta poslovna logika ne smije biti dio prikaza jer se on bavi isključivo prikazom podataka, dok model zahvaljujući labavoj sprezi jedini sadržava glavnu logiku.

4.3.3. Modeli

Model je jedan od tri glavna sloja ASP.MVC predloška u kojem nalazimo glavnu poslovnu logiku i zapise koji se pohranjuju u bazi podataka (eng. *database*) zbog toga se on smatra najvažnijim dijelom MVC arhitekture. Modeli se mogu sastojati od jednog ili pak više kolekcija objekata, jedino ne smiju sadržavati detalje implementacije. Model se najčešće sastoji od razreda koji podatke prikazuju kao svojstva i logiku u obliku metoda. Spomenute klase su različitih veličina i oblika, a obično dolaze kao „model podataka“ ili „model domene“ koji je zadužen za upravljanje podacima [4]. Zato kažemo da modeli predstavljaju domenu na koju se aplikacija fokusira. U ASP.NET MVC okviru modele smatramo objektima koji služe za slanje informacija bazi podataka, izvođenje poslovnih računica, pa čak i za generiranje prikaza što je prikazano u kodu 2. Mehanizam Entity Framework je neizostavni dio MVC aplikacije, a služi za pohranu .NET objekata i dohvat istih uz pomoć LINQ (engl. *Language Integrated Query*) upita. LINQ je komponenta .NET Framework-a koja dodaje standardne operatore za zadavanje upita .NET jezicima poput C# ili VB.NET.

Kod 2. Klasa modela za pohranu dnevnice u bazu podataka

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;

namespace WebAplikacijaMvc.Models
{
    public class Dnevnica
    {
        [Required(ErrorMessage = "Unesite mjesto polaska.")]
        [Display(Name = "Polazište")]
        public string Polaziste { get; set; }

        [Required(ErrorMessage = "Unesite mjesto
odredišta.")]
        [Display(Name = "Odredište")]
        public string Odrediste { get; set; }

        [Required(ErrorMessage = "Unesite vrijeme polaska.")]
        [Display(Name = "Vrijeme polaska")]
        public string VrijemePolaska { get; set; }

        [Required(ErrorMessage = "Unesite vrijeme dolaska.")]
        [Display(Name = "Vrijeme dolaska")]
        public string VrijemeDolaska { get; set; }
    }
}
```

5. OFFICE 365 PLATFORMA

Office 365 platforma je zapravo skupina Microsoft Office aplikacija koje su korisnicima dostupne putem Clouda, dok je službeni logo prikazan na slici 5. Sastoji se od više servisa, a neki od njih su Exchange Online za slanje elektroničke pošte, Lync Online za komunikaciju te Office Web Apps koji se sastoji od tradicionalnih Office aplikacija [8]. Jedna od velikih prednosti je ta da se aplikacije redovito ažuriraju otprilike svaka tri mjeseca s time da pretplatnici ne moraju plačati nadogradnju. Platforma Office 365 korištena je za funkcionalnost slanja elektroničke pošte u slučaju da korisnik aplikacije nije unio dnevnicu u bazu podataka. E-mail se u tom slučaju generira i šalje uporabom servisa Microsoft Exchange 2013.

Slika 5. *Office 365 platforma*

(http://www.skypeassets.com/content/dam/scom/images/text-and-image/mobile/office-logo_v3.jpg)

5.1. Servis Microsoft Exchange 2013

Exchange je Microsoftov mail server account koji služi za slanje elektroničke pošte putem Microsoft Outlook-a i njegov logo je prikazan na slici 6. Koristi se među mnogim poduzećima jer napredna svojstva Outlook-a zahtjevaju korištenje specifične verzije Exchange računa [9]. U ovom slučaju, u sklopu Office platforme, korišten je Microsoft Exchange Server 2013 i njegov Autodiscover servis koji je potreban za automatsko konfiguriranje korisničkih postavki Outlook klijenata i pristup Exchange značajkama.

Slika 6.*Microsoft Exchange Server 2013*

(http://aide.hosteur.com/data/uploads/microsoft_exchange.jpg)

Dio sustava koji predstavlja glavnu metodu za slanje mailova sa parametrima primatelja, sadržajem i naslovom poruke prikazan je na kodu 3. Na taj način je sustav osposobljen za automatizirano kreiranje univerzalne HTML poruke koju se može poslati zahvaljujući integraciji sa Office 365 platformom. Metoda se poziva samo na poziv Azure Schedulera tokom provjere unosa u bazu i to ukoliko dnevnica pojedinog korisnika nije pronađena u bazi podataka.

Kod 3. Slanje e-maila korištenjem Office 365 platforme

```
private static void sendExchangeServiceEmail(string Receiver,
string Body, string Subject)
     {
                ExchangeService service = new
ExchangeService(ExchangeVersion.Exchange2013);
                service.Credentials = new
WebCredentials("mrisek@msn.com", "Lozinka");
                service.Url = new
Uri("https://outlook.office365.com/EWS/Exchange.asmx");
                service.AutodiscoverUrl("mrisek@msn.com",
                     RedirectionUrlValidationCallback);

                EmailMessage message = new
EmailMessage(service);
                message.Subject = Subject;

                var MailBody = new MessageBody(BodyType.HTML,
Body);
                message.Body = Body;
                message.ToRecipients.Add(Receiver);
                message.SendAndSaveCopy();
     }
```

Ako dođe do tog scenarija, inicijalizira se novi ExchangeService objekt i odabire odgovarajuća verzija servisa. To su najvažnije linije koda u kojima je ostvarena integracija Office 365 platforme i MVC aplikacije. Korisnički podaci za pristup servisu u obliku e-mail adrese korisnika i lozinke zapisani su u Web.config datoteci MVC aplikacije i koriste se za autorizaciju. Uri platforme Office 365 i metoda za validaciju Url-a su također potrebni za uspješnu realizaciju servisa.

Kod 4. Validacija Url-a korištenjem Exchange 2013 servisa

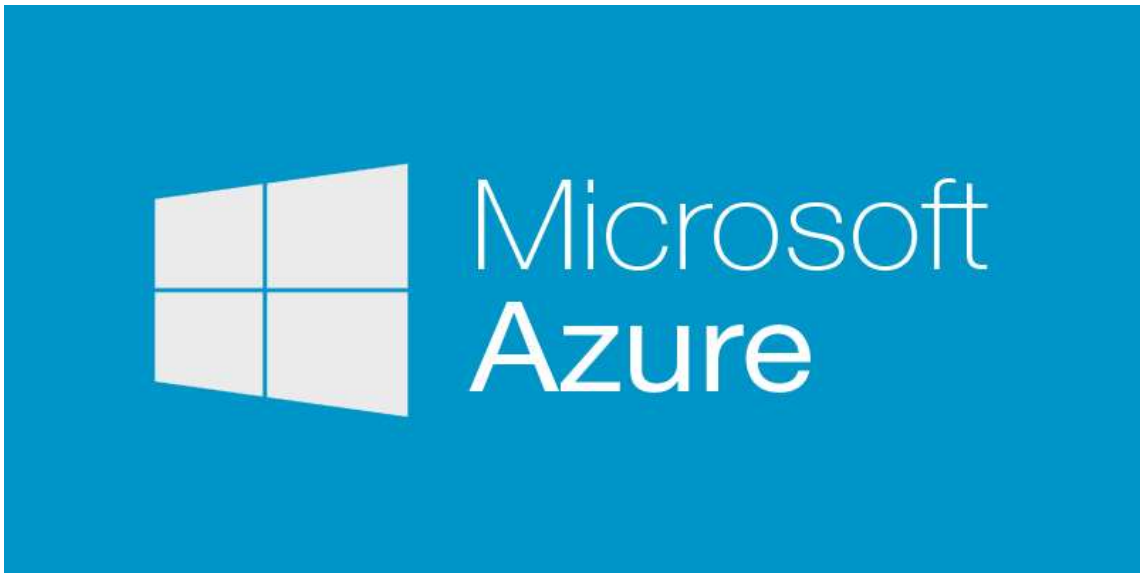
```
private static bool RedirectionUrlValidationCallback(String
redirectionUrl)
        {
            bool redirectionValidated = false;
            if (redirectionUrl.Equals(
                "https://autodiscover-
s.outlook.com/autodiscover/autodiscover.xml"))
                redirectionValidated = true;

            return redirectionValidated;
        }
```

Na kodu 4 se vidi još jedan neophodan segment Exchange Servera, a to je prisustvo Autodiscover servisa koji služi za validaciju korisničkih postavki s ciljem pristupa Exchange servisu [10]. Tek nakon te provjere omogućeno je slanje elektroničke pošte svakom korisniku sustava. Najprije se inicijalizira novi EmailMessage objekt kroz Microsoft Exchange servis i zatim se definira naslov e-maila, sadržaj, primatelj te eventualni privici da bi se napokon poslao e-mail i spremila kopija među poslane poruke.

6. MICROSOFT AZURE PLATFORMA

Microsoft Azure je cloud platforma sa golemom kolekcijom integriranih servisa koji pokrivaju razna područja od računarstva, analitike, baza podataka, mreža i slično. Koristi se diljem svijeta za izgradnju, hosting i upravljanje aplikacijama, a njegov prepoznatljiv logo prikazan je na slici 7. Podržava mnoge operacijske sustave, programske jezike i uređaje te koristi pouzdanu tehnologiju u Paas (engl. *Platform as a service*) i IaaS (engl. *Infrastructure as a service*) servisima. Sadrži mnoštvo integriranih alata, predložaka i servisa koji olakšavaju vođenje mobilnih, IoT (engl. *Internet of Things*) i web aplikacija [11]. Azure platforma pruža razne mogućnosti integracije sa postojećim IT okruženjima, pa su zato u ovom radu upotrebljeni neki od njezinih najpoznatijih servisa poput Web App-a za deployment aplikacije, SQL Database-a za upravljanje bazom podataka i Scheduler-a za automatiziranu provjeru unosa dnevnica u sustav.

Slika 7. *Logo Microsoft Azure servisa*



(http://kemptechnologies.com/blog/wp-content/uploads/2014/07/micorosftazurelogo.jpg)

6.1. Azure Web App

Azure Web App servis je multifunkcionalni servis koji služi za postavljanje aplikacija na Azure Cloud, te se pomoću njega mogu razvijati aplikacije za bilo koji uređaj ili platformu. Servis stoga sadrži bogat spektar rješenja za web i mobilne aplikacije uz mogućnost raznih integracija. Jednostavan je za upotrebu, pouzdan i siguran, te uvelike olakšava automatizaciju poslovnih procesa [12]. MVC 5 aplikacija je postavljena na Cloud uz pomoć Azure Web App servisa što je omogućilo postavljanje (engl. *deployment*) cijelog projekta direktno iz razvojnog okruženja Microsoft Visual Studio 2013, ali i integraciju sa Azure SQL bazom podataka u kojoj su zapisani svi korisnici sustava i njihovi unosi. Dodatna mogućnost koju nudi Web App servis je Visual Studio Online pomoću kojeg je direktno iz internet preglednika moguće vidjeti sve datoteke, upravljati njima i mijenjati ih. Osim toga, servis nudi mnogo statističkih podataka vezano uz samu aplikaciju koji služe za praćenje brzine i efikasnosti aplikacije.

6.2. Azure SQL Database

SQL Database je servis koji pruža mogućnost korištenja relacijskih baza podataka putem Cloud-a. U potpunosti je kompatibilan sa MVC aplikacijama i Web App servisom, a temelji se na Microsoft SQL Server DBMS-u (engl. *Database management system*). Podržava sve postojeće SQL Server alate, knjižnice i API-je (engl. *Application programming interface*), jednostavan je za upotrebu te je moguće odabrati idealan paket usluga ovisno o potrebama aplikacije [13]. Zaštita podataka realizirana je pomoću kontrole i obnove podataka iz transakcija koje su provedene tokom posljednjih 35 dana. U ovom projektu je SQL Database servis služio za pohranu svih korisničkih podataka, za unos dnevnica u sustav preko MVC 5 aplikacije te za provjeru unosa u aplikaciju.

Slika 8. *Logo Microsoft Azure SQL baze podataka*

(http://blogs.technet.com/blogfiles/dataplatforminsider/WindowsLiveWriter/MicrosoftSQLServicesisnowMicrosoftSQLAzu_958F/SQL-Azure_rgb_2.png)

6.3. Azure Scheduler

Microsoft Azure Scheduler je servis koji služi za pokretanje definiranih zadataka u jednostavnijem ili složenijem vremenskom rasporedu. Koristi se za pozivanje servisa ili skripti neovisno o tome nalazi li se ta web stranica unutar Microsoft Azure-a ili ne. Potrebno je samo kreirati akciju, tj. zadatak (eng. *Job*) koji će se pozivati odmah, u određenom trenutku u budućnosti ili će se pak ponavljati na dnevnoj bazi. Ovaj pouzdani servis je idealan za akcije koje se često ponavljaju ili pak izvršavaju na dnevnoj bazi te za pokretanje asinkronih procesa [14]. Točno vrijeme izvršavanja, broj ponavljanja i trajanje zadatka moguće je brzo i jednostavno podesiti kroz Azure Portal pa je ta funkcionalnost zbog toga iskorištena u svrhu pokretanja provjere unosa dnevnica u sustav. Servis je upotrijebljen na način da se u točno određeno vrijeme pozove napisana skripta koja pokrene provjeru. Za potrebe ovog projekta, idealno je bilo izvršiti kontrolu svakog radnog dana u ponoć, nakon čega aplikacija svakog korisnika koji nije unio dnevnicu za prethodni dan obavijesti o tome putem elektroničke pošte koja se potom šalje pomoću Exchange servisa iz Office 365 platforme.

7. RAZVOJ APLIKACIJE

Svjedoci smo golemog napretka Internet stranica koji je ostvaren tokom posljednjih godina. Nekad su se klasične web stranice sastojale samo od JavaScripta i HTML koda, međutim, danas korisnici očekuju neusporedivo više od popularnih internet aplikacija. Gmail, Twitter, Facebook i sl. visoko su podigli ljestvicu korisničkih zahtjeva pa sada web aplikacije sve više nalikuju na bogate stolne aplikacije. Korisnike više ne privlači jednostavni tekst pa tako dolazi do stalnog razvitka u području najnovijih Internet preglednika i usvajaju se novi standardi poput HTML 5 ili CSS 3 na kojima će se u budućnosti bazirati još interaktivnije i kompleksnije aplikacije. Isto tako, ASP.NET MVC arhitektura otišla je korak naprijed prema stvaranju bogatih aplikacija olakšanim programiranjem na strani klijenta uz pomoć biblioteke jQuery. Osim toga pojednostavnjena je animacija i obrada događaja te ubrzan rad aplikacija upotrebom Ajax tehnologije. Taj razvoj je kod MVC platforme započeo dodavanjem jQuery biblioteke u projektne predloške od izdanja ASP.NET MVC 3.

Slika 9. *HTML 5 i CSS 3 logo*

(http://geteverything.org/wp-content/uploads/2013/08/13.png)

7.1. JavaScript

JavaScript je objektno-orijentirani skriptni jezik ili programski jezik HTML-a i web stranica, a njegov logo je prikazan na slici 10. Jednostavan je, ne ovisi o platformi, povezuje se s objektima na klijentu i služi za njihovo upravljanje. Razvila ga je tvrtka Netscape, a JavaScript standard naziva se ECMAScript. Od 2012. godine svi moderni internet preglednici u potpunosti podržavaju ECMAScript 5.1 ili barem ECMAScript 3 [5]. Ovaj dinamični skriptni jezik se bazira na prototipovima i može se upotrijebiti za programiranje reakcija određene web stranice u slučaju nekog događaja (engl. *event*). Sintaksa JavaScripta je slična programskim jezicima C++ i Java što pojednostavnjuje njegovo učenje. Jedna od najpoznatijih tehnologija koju koristi ovaj JavaScript standard u ASP.MVC platformi zove se Ajax.

Slika 10. *Logo skriptnog programskog jezika JavaScript*

(http://techieaddict.com/wp-content/uploads/2015/03/JavascriptLogo-navsingh.org_.uk_.png)

7.2. Ajax

Ajax (engl. *Asynchronous JavaScript and XML*) u doslovnom prijevodu znači asinkroni JavaScript i XML, a riječ je o važnoj internet tehnologiji koja omogućava JavaScriptu komunikaciju sa web serverom. Ajax služi za slanje i primanje podataka u raznim formatima, bilo da se radi o HTML, CSS, DOM, XSLT ili JSON tehnologijama, a njegov logo je prikazan na slici 11. Zahvaljujući Ajaxu i asinkronoj kombinaciji navedenih tehnologija na nekoj internet stranici se mogu dinamički prikazivati novi sadržaji bez ponovnog pokretanja cijele stranice. Isto tako treba napomenuti da je web aplikacija koja koristi Ajax vizualno impresivnija jer korisničko sučelje brže reagira na akcije korisnika [6]. ASP.MVC 5 je moderan okvir za razvoj web aplikacija koji ima podršku za Ajax od samog stvaranja projekta, a najveći dio tih svojstava baziran je na JavaScript biblioteci otvorenog koda koja se naziva jQuery.

Slika 11. *Asynchronous JavaScript and XML logo*

(http://www.webkingtechnologies.com/wp-content/uploads/2010/12/ajax_web_development.jpg)

7.3. jQuery

jQuery je jedna od najpopularnijih JavaScript biblioteka današnjice, a služi za olakšanu upotrebu JavaScripta. Otvorenog je koda i koristi se za stvaranje animacija i dodavanje interaktivnosti web stranicama. Ovu biblioteka možda najbolje opisuje njezin moto „Piši manje, radi više“, koji je prikazan na slici 12, jer je lagan za učenje i uvelike olakšava upravljanje HTML-om ili Ajaxom uz uvjet poznavanja skriptnog jezika JavaScript. Jednostavan i prilagodljiv API (engl. *application programming interface*) podržan je od strane svih modernih internet preglednika današnjice (Firefox, Chrome, Internet Explorer, Opera, Safari i tako dalje) i uspješno se koristi za sakrivanje njihovih grešaka i nedosljednosti [7]. jQuery se danas također nalazi u ASP.MVC aplikacijama, a projektni predložak će automatski postaviti sve potrebne datoteke u direktorij s nazivom „Scripts“ prilikom stvaranja novog projekta. Unutar MVC 5 okvira, jQuery skripte dodane su pomoću NuGeta, što garantira brzo i jednostavno ažuriranje aktualne verzije čim se ona pojavi. Zahvaljujući jQuery-ju, MVC platforma pruža funkcionalnosti poput validacije na strani klijenta i asinkronog slanja podataka.

Slika 12. *Moto jQuery biblioteke je „Piši manje, radi više.“*

(http://3.bp.blogspot.com/-L0lal12jkwU/VYjurU-Mo1I/AAAAAAAAEwQ/NaJi3tVDDhk/s640/jquery_logo.png)

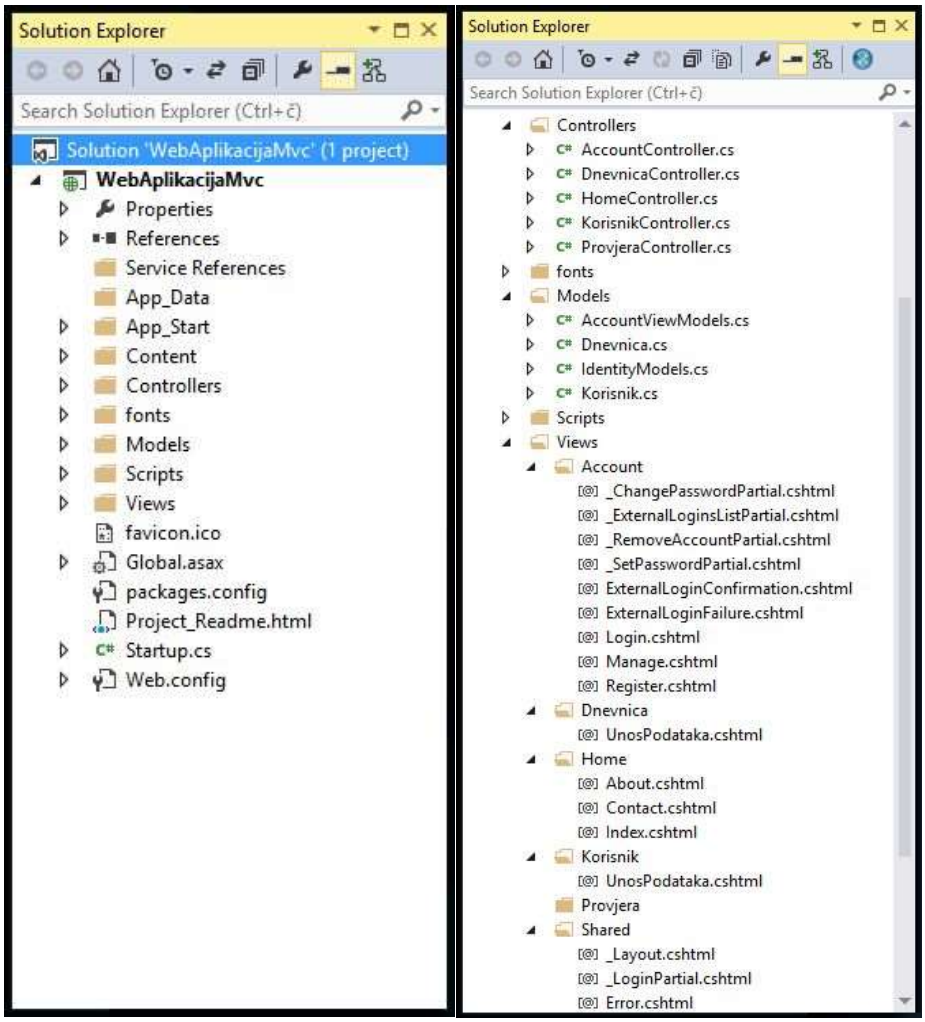
8. MVC APLIKACIJA

Aplikacija služi za unos dnevnica u bazu podataka te ima ugrađenu provjeru unosa istih u sustav. Korisnici koji nisu napravili unos tog dana, biti će o tome obavješteni putem e-maila na njihovu adresu. Riječ je o ASP.NET MVC aplikaciji postavljenoj na Microsoft Azure servis kao Web App čija se provjera unosa pokreće svaki radni dan u zadano vrijeme u sklopu Job-a kreiranog pomoću Azure Schedulera. Za provjeru unesenih dnevnica u ovu ASP.MVC web aplikaciju zadužen je integrirani Exchange servis iz Office 365 platforme.

Mapa projekta aplikacije ovog Završnog rada prikazana je na slici 13. Sastoji se od više glavnih dijelova pa se stoga tu nalaze svojstva, reference i ostale datoteke sortirane po direktorijima. U XML datoteci PublishProperties pod svojstvima nalaze se sve potrebne informacije za uspješno postavljanje projekta na server, dok su u datoteci AssemblyInfo.cs sadržane značajke projekta poput opisa, verzije, konfiguracije i slično. Pod References možemo locirati sve .dll datoteke koje su potrebne za ispravan rad aplikacije, a moguće je dodati i vanjsku referencu u Service References koji će služiti za dohvat podataka iz baze. App_Data je direktorij gdje će aplikacija spremati log file ili neki drugi tip datoteke prema potrebi. Klasa za upravljanje zahtjevima prema serveru nalazi se u BundleConfig.cs datoteci koja je smještena u App_Start direktoriju. Tamo je također moguće dodati filtere, konfigurirati url putanje, te uključiti kolačiće ili autentifikaciju, a sve od navedenog će se izvršiti prilikom pokretanja aplikacije.

U datoteci Content nalazi se poddirektorij naziva Images u kojem možemo pronaći sve slike koje se koriste u web aplikaciji, a osim toga ovdje su također smještene i glavne CSS datoteke kojima je zadan dizajn aplikacije. U fonts folderu nalaze se sve datoteke potrebne za oblikovanje fontova aplikacije, a unutar Scripts direktorija smještene su jQuery i Bootstrap javascript datoteke. Od preostalog sadržaja projekta valja istaknuti glavnu konfiguracijsku datoteku Web.config i direktorije Controllers, Models i Views.

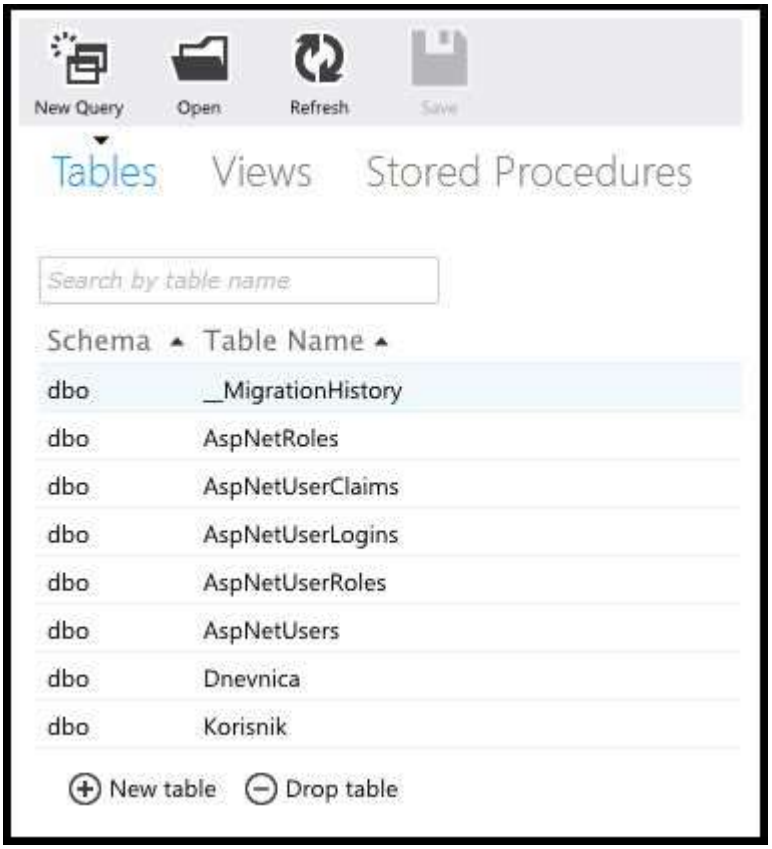
Slika 13. *Mapa projekta i struktura pristupa Model-Prikaz-Kontroler*



8.1. Baza podataka

SQL Database Management Portal je dio SQL Database servisa koji služi za pregled, administraciju i dizajniranje baze podataka putem Azure Platforme. Pristup podacima ima samo ovlašteni korisnik koji najprije mora unijeti korisničko ime i lozinku za taj server. Nakon toga može neometano formirati upite prema bazi podataka, izraditi pohranjene procedure (engl. *Stored Procedures*) ili dodijeliti ovlaštenja upotrebom pogleda (engl. *View*) kao mehanizma zaštite što je i prikazano na slici 14.

Slika 14. *Meni SQL Database Management Portala i tablice baze podataka*



Model baze podataka je izrađen uz pomoć programa Microsoft SQL Management Studio, sastoji se od osam tablica prikazanih EER dijagramom (engl. *Extended Entity Relations Diagram*) na slici 14. koje su tijekom postavljanja na Azure server povezane sa Microsoftovim servisom SQL Database pomoću stringa za povezivanje (engl. *connection string*). Kompletni podaci iz svih relacija su pohranjeni na serveru s ciljem kontrole nad njima.

Tablica AspNetRoles nam pomaže prilikom autorizacije, što znači da služi za određivanje posebnog sadržaja kojem određeni korisnik smije pristupiti. Zbog toga su u toj tablici sadržani nazivi svih uloga(engl. *roles*) koje je moguće dodijeliti korisnicima. AspNetRoles tablica sastoji se od polja s nazivima rola i njihovim identifikacijskim brojevima.

Tablica AspNetUserClaims također služi za provedbu autorizacije jer se koristi prilikom dodjeljivanja prava pristupa korisnicima web aplikacije. Tako će korisniku odmah

prilikom dodavanja neke role automatski biti pridjeljena i sva prava definirana za tu ulogu. AspNetUserClaims tablica sastoji se od polja s tipom prava, vrijednošću prava, korisničkog ID-ja te identifikacijskog broja.

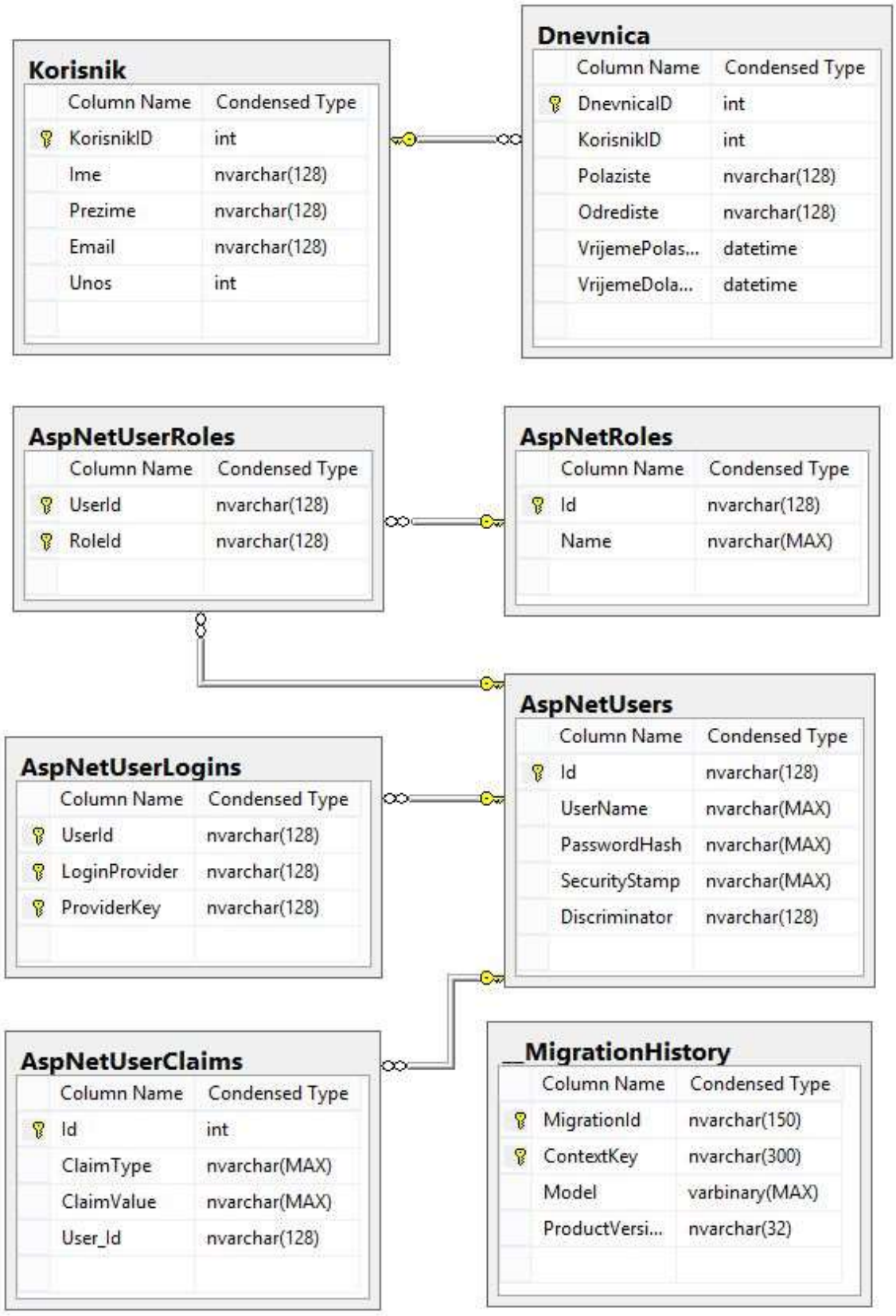
Tablica AspNetUserLogins koristi se prilikom logiranja, a služi za omogućavanje prijave u sustav putem vanjskih pružatelja usluga. Namjena te tablice je provjeriti postoji li alternativni način prijave preko akcijskih metoda ugrađenih u ASP.MVC aplikaciju i polja koje sadrže identifikacijsku vrijednost, naziv vanjskog pružatelja usluga te univerzalni ključ.

Tablice AspNetUserRoles i AspNetUsers također imaju primjenu kod provedbe autorizacije. U tablicu AspNetUserRoles se spremaju identifikacijski brojevi korisnika i uloge dodjeljene tom korisniku, a u tablicu AspNetUsers pohranjujemo ID, korisničko ime, hash vrijednost lozinke, SecurityStamp i Discriminator. Security Stamp je zapravo token koji služi za provjeru stanja korisničkog računa kao i Discriminator, dok PasswordHash vrijednost služi za validaciju lozinke.

Tablica _MigrationHistory služi za pohranu svih promjena koje su napravljene u bazi podataka. Vrijednosti koje se pohranjuju u tablicu _MigrationHistory su MigrationId, ContextKey, Model i ProductVersion, a spomenutu tablicu je moguće prilagoditi ovisno o potrebama.

Osim nabrojenih sistemskih tablica, baza podataka se sastoji i od dvije tablice koje sadrže poslovnu logiku. Te tablice kreiraju se u skladu sa zahtjevima klijenta, a u ovom Završnom radu napravljene su tablice Dnevnica i Korisnik koje su prikazane u sklopu modela na slici 15. Kako se radi o aplikaciji za unos dnevnica u sustav, tablica Dnevnica sadrži polja u koja se zapisuju polazište, odredište, vrijeme polaska i dolaska, dok se u tablicu Korisnik spremaju ime, prezime, e-mail adresa te boolean vrijednost izvršene provjere unosa dnevnice.

Slika 15. *EER dijagram baze podataka*



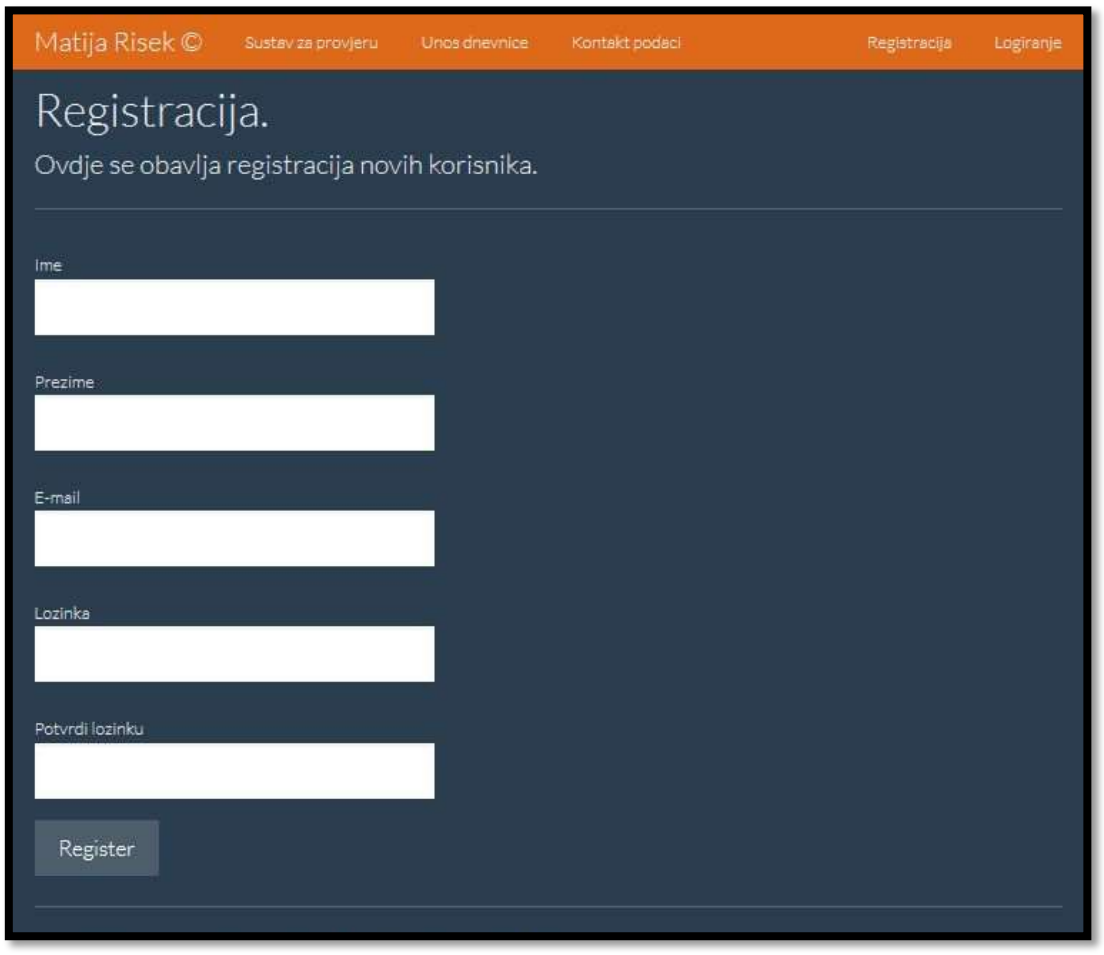
8.2. Web aplikacija

U skopu ovog Završnog rada razvijena je web i mobilna aplikacija, a obje se sastoje od nekoliko osnovnih modula:

- Modul za registraciju i logiranje,
- Modul početne stranice sustava,
- Modul za unos dnevnice,
- Modul za prikaz informacija.

Registrirani korisnici imaju pristup svim stranicama sustava, dok neregistrirani korisnici mogu pristupiti jedino modulu za registraciju i logiranje.

Slika 16. *Modul za registraciju novog korisnika*



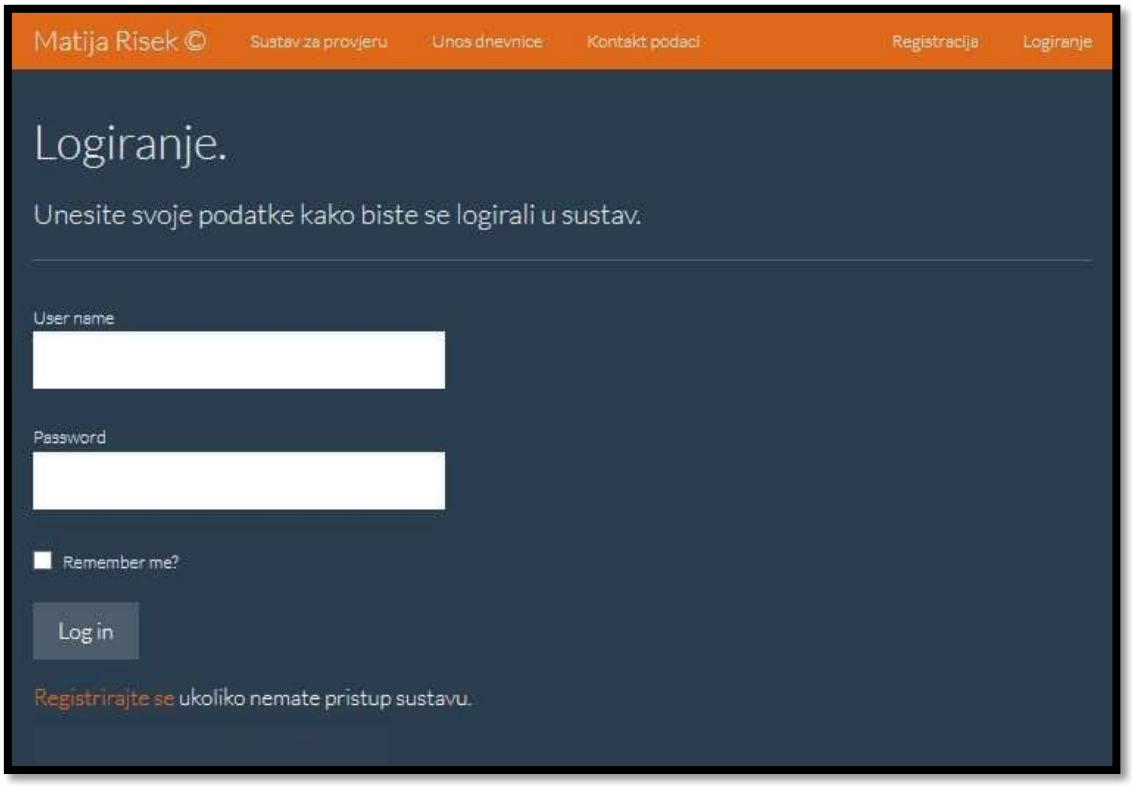
Modul za registraciju novih korisnika dostupan je svim korisnicima sustava, koristi se za kreiranje novog korisničkog računa, a prikazan je na slici 16. Jednom kreiran, korisnički račun služi za autorizaciju u svrhu pristupa ostalim modulima aplikacije. U ovom slučaju, registrirani korisnik moći će nakon uspješnog logiranja pristupiti modulu početne stranice sustava, modulu za unos dnevnice i modulu za prikaz informacija. Nakon popunjavanja svih polja i pritiska na tipku „Registracija“, upisani podaci se validiraju i potom zapisuju u Azure SQL bazu podataka. Memorirani podaci će se nakon toga provjeravati nakon svake odjave iz sustava u sklopu autorizacije iz modula za logiranje. Sve unesene podatke, pa čak i lozinku, registrirani korisnik može izmjeniti u bilo kojem trenutku i to preko dodatne stranice za upravljanje korisničkim računom.

Kod 5. Primjer poziva jQuery validacije

```
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Forma za unos podataka je realizirana kroz akcijsku metodu Register() kontrolera Account i Post metode koja služi za slanje podataka. MVC aplikacija generira token za validaciju prilikom slanja forme kroz AntiForgeryToken() metodu što služi za zaštitu ranije spomenute akcijske metode. Povratne informacije ispisuju se na mjestima pogrešnog unosa u formi putem ValidationSummary() metode. Validacija na kontrolama formi za unos podataka se pokreće uz pomoć Scripts.Render() metode koja zatim poziva jQuery provjeru i to se vidi na kodu 5, dok se identitet korisnika utvrđuje Identity.GetUserName() metodom.

Slika 17. *Modul za prijavu postojećih korisnika u sustav*



Modul za prijavu postojećih korisnika u sustav služi za omogućavanje pristupa modulu početne stranice, modulu za unos dnevnice i modulu za informacije. Dostupan je svima i sastoji se od forme za unos podataka gdje se upisuju korisničko ime i lozinka ranije kreiranog korisničkog računa što je prikazano na slici 17. Ukoliko korisnički račun za pristup sustavu još nije kreiran, može se napraviti pritiskom na „Registrirajte se“ ili preko navigacije, nakon čega će korisnik biti preusmjeren na modul za registraciju novog korisnika. Za uspješno logiranje u sustav potrebno je popuniti sva obavezna polja forme i pritisnuti tipku „Logiranje“. Nakon toga korisnik će automatski biti preusmjeren na početnu stranicu aplikacije ukoliko prođe postupak provjere. U suprotnom, može probati ponoviti upis podataka iz korisničkog računa.

Kod 6. Klasa LoginViewModel

```
public class LoginViewModel
{
      [Required]
      [Display(Name = "Korisničko ime")]
      public string UserName { get; set; }

      [Required]
      [DataType(DataType.Password)]
      [Display(Name = "Lozinka")]
      public string Password { get; set; }

      [Display(Name = "Zapamti me?")]
      public bool RememberMe { get; set; }
}
```

Prilikom obrade podataka unesenih u formu za prijavu registriranih korisnika, MVC aplikacija se služi modelom LoginViewModel, čija klasa je prikazana na kodu 6. Spomenuti model zna koja su polja obavezna i raspoznaje podatke zahvaljujući točno definiranim atributima, pa zato tekst lozinke automatski sakriva zvjezdicama, prepoznaje boolean vrijednost checkboxa „Zapamti me“, a tekstualni unos korisničkog imena ostavlja nedirnutim na prikazu.

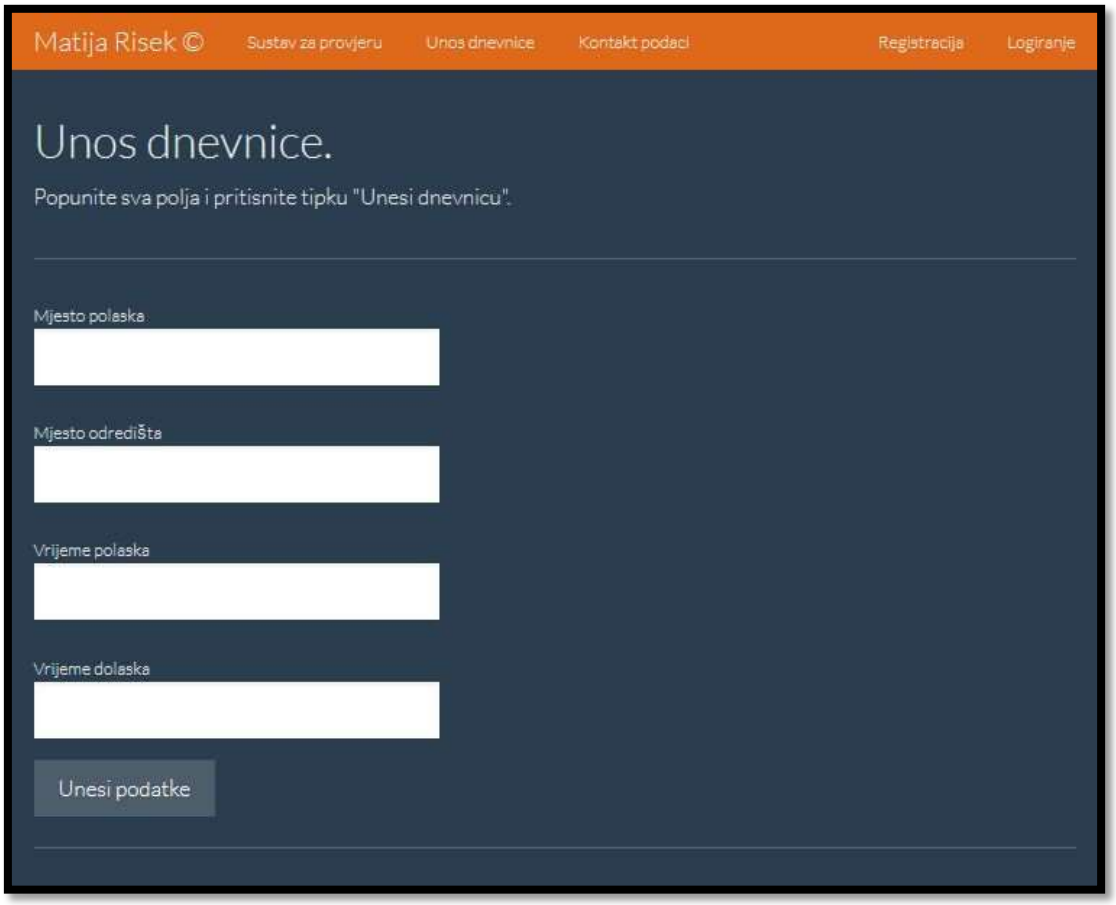
Slika 18. *Modul početne stranice sustava*

Nakon uspješne prijave u sustav, korisnik je automatski preusmjeren na početnu stranicu web aplikacije koja se zove „Sustav za provjeru“. Pristup toj stranici nije dopušten neregistriranim korisnicima, a ovlašteni korisnik odavde ima pristup svim preostalim modulima sustava preko navigacije. Uz pomoć Bootstrapa napravljen je web dizajn koji se automatski prilagođava bilo kojem uređaju na kojem je pokrenut, bez obzira radi li se o tabletu, pametnom telefonu ili stolnom računalu. Na slici 18. je prikazan izgled ovog modula na zaslonu tableta. Naslov na istom modulu dinamički je kreiran korištenjem objekta ViewBag kao jednostavan primjer odvajanja logike od prikaza kod ASP.MVC s ciljem agilnog razvoja i pune kontrole nad web aplikacijom. Na sličan način je izvedena navigacija aplikacije preko zasebnog prikaza koji dinamički generira anchor elemente ovisno o tome je li korisnik prijavljen u sustav. Taj postupak je realiziran uporabom sintakse Razor i prikazan je u kodu 7.

Kod 7. Razor sintaksa i dinamičko kreiranje navigacije

```
@using Microsoft.AspNet.Identity
@*Slučaj 1. Korisnikov identitet je autoriziran*@
@if (Request.IsAuthenticated)
{
    using (Html.BeginForm("LogOff", "Account",
FormMethod.Post, new { id = "logoutForm", @class = "navbar-
right" }))
    {
        @Html.AntiForgeryToken()
        <ul class="nav navbar-nav navbar-right">
            <li>
                @Html.ActionLink("Pozdrav, " +
User.Identity.GetUserName() + "!", "Manage", "Account",
routeValues: null, htmlAttributes: new { title = "Manage" })
            </li>
            <li><a
href="javascript:document.getElementById('logoutForm').submit
()">Odjava</a></li>
        </ul>
    }
}
@*Slučaj 2. Identitet korisnika nije autoriziran*@
else
{
    <ul class="nav navbar-nav navbar-right">
        <li>@Html.ActionLink("Registracija", "Register",
"Account", routeValues: null, htmlAttributes: new { id =
"registerLink" })</li>
        <li>@Html.ActionLink("Logiranje", "Login", "Account",
routeValues: null, htmlAttributes: new { id = "loginLink"
})</li>
    </ul>
}
```

Slika 19. *Modul za unos dnevnice u bazu podataka*



Na slici 19. prikazan je modul za unos dnevnice koji registriranom korisniku omogućuje upisivanje vrijednosti značajne za poslovnu logiku aplikacije. Korisnik tako može u formu unijeti podatke poput mjesta polaska, mjesta odredišta i odrediti točno vrijeme polaska i dolaska preko Bootstrap datepickera koji je podržan u MVC aplikaciji. Nakon pritiska na tipku „Unesi podatke“ aplikacija se spaja na Azure SQL Database i unosi zapise u odgovarajuću tablicu relacijske baze podataka. U ponoć se aktivira provjera i obavještava se svakog korisnika koji nije poslao podatke za prethodni dan. Izgled i sadržaj forme moguće je prilagoditi različitim tvrtkama ovisno o njihovim potrebama.

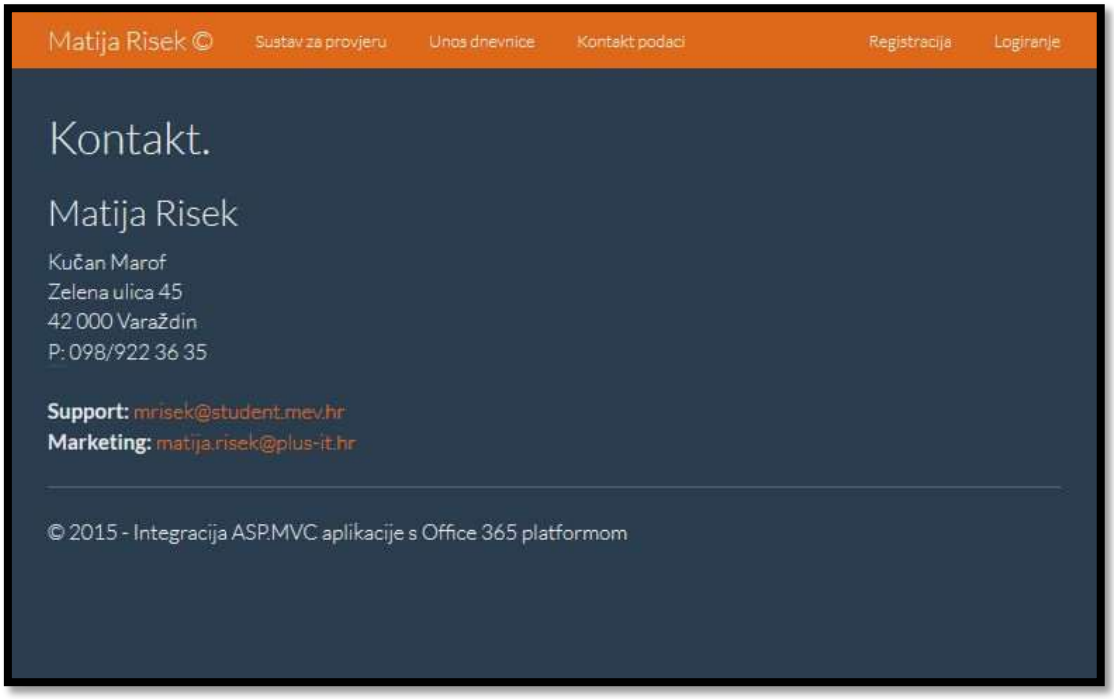
Kod 8. Unos dnevnice u bazu podataka

```
public ActionResult
UnosPodataka(WebAplikacijaMvc.Models.Dnevnica model){
            SqlConnection con = new
SqlConnection(WebConfigurationManager.AppSettings["SqlDatabas
eConnectionString"]);
            SqlCommand cmd = new SqlCommand();
            cmd.Connection = con;

            cmd.CommandText = "insert Dnevnica values((select
KorisnikID from korisnik where KorisnickoIme = '"
                + User.Identity.GetUserName() + "'), '"
                + model.Odrediste + "', '"
                + model.Polaziste + "', '"
                + model.VrijemePolaska + "', '"
                + model.VrijemeDolaska + "');";

            con.Open();
            cmd.ExecuteNonQuery();
            con.Close();
}
```

Rezultati akcijskih metoda kod MVC okvira mogu vratiti određeni tip podataka. Primjer metode zadužene za stvaranje i pohranu novog objekta Dnevnica koji je definiran u modelu aplikacije prikazan je u kodu 8. Na početku metode dohvaća se string za spajanje na Sql Server bazu podataka iz Web.config datoteke na temelju kojeg se inicijalizira objekt za otvaranje veze s bazom podataka. Zatim se definira novi unos u bazu upotrebom SqlCommand klase pomoću kojeg aplikacija prikuplja sve podatke unesene u formu i sprema iste u odgovarajuće relacije. Vrijedi istaknuti kako aplikacija dinamički određuje identitet aktivnog korisnika kroz funkciju GetUserName() koja dohvaća korisničko ime te vrši upit u tablicu „Korisnik“. Svakoj obrađenoj dnevnici je zahvaljujući tome pridružen jedinstveni identifikacijski broj korisnika koji je istu unio u

sustav. Preostali podaci iz forme za unos dnevnica se također dohvaćaju nakon čega slijedi naredba za otvaranje veze sa bazom podataka. Tek tada počinje izvršavanje zadanog upita naredbom ExecuteNonQuery(), vraća se broj redaka na koje je naredba imala utjecaj, te se konačno zatvara veza metodom Close().

Slika 20. *Modul za prikaz kontakt informacija*



Modul za prikaz informacija služi isključivo za ispis telefonskih brojeva ili e-mail adresa koje je potrebno istaknuti jer su značajne za kontekst aplikacije, a jedan takav primjer pokazan je na slici 20. Pritiskom na neku od istaknutih adresa aktivira se Microsoft Outlook spreman za slanje elektroničke pošte na zadanu adresu. Kao i na ostalim modulima, korisniku je omogućen prelazak između modula putem navigacije, dok se dizajn jednostavno prilagođava neovisno o uređaju na kojem je web aplikacija pokrenuta. Za univerzalan dizajn ovog i svih ostalih modula korišten je Bootstrap u kojeg je implementirana nadograđena Bootswatch open-source tema Superhero.

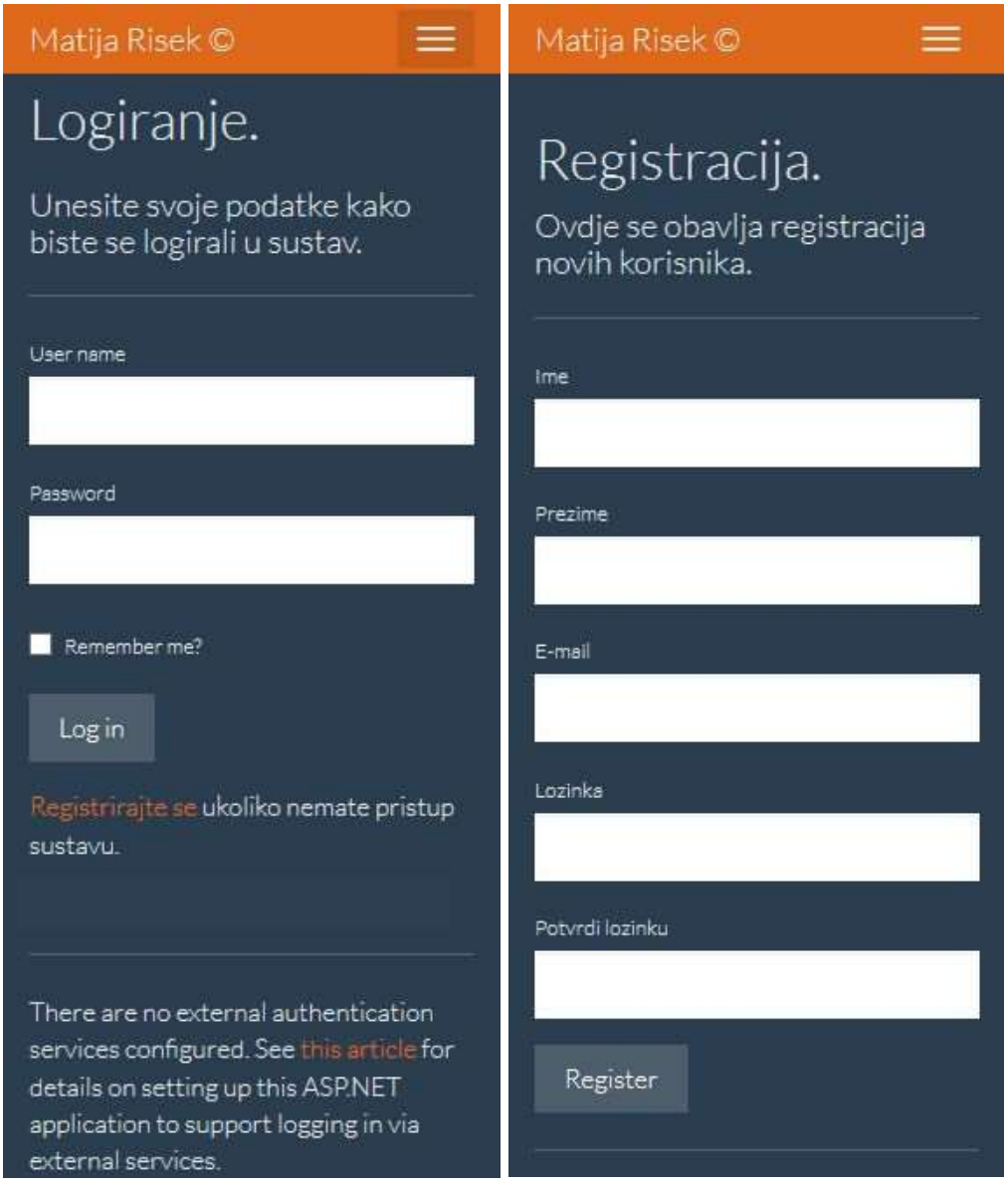
8.3. Mobilna aplikacija

Moduli početne stranice i moduli za unos dnevnice sadrže iste funkcionalnosti kod mobilne verzije kao i na web komponenti što se vidi na slici 21. Prvi služi kao početna stranica nakon uspješnog procesa autentifikacije korisnika poslije logiranja u sustav, a drugi za popunjavanje forme s ciljem unosa dnevnice u bazu podataka. Vrijedi naglasiti kako se, zahvaljujući Bootstrapu, dizajn mobilne aplikacije u potpunosti prilagodio veličini ekrana mobilnog uređaja.

Slika 21. *Prikaz modula početne stranice i modula za unos dnevnice*

Moduli za registraciju i logiranje služe za popunjavanje formi sa korisničkim podacima u svrhu registracije odnosno autentifikacije prilikom prijave u sustav baš kao i u internetskoj komponenti. Korisnik može odabrati checkbox „Zapamti me“ pa u tom slučaju idući put neće morati upisivati podatke korisničkog računa prilikom idućeg logiranja. Navedeni moduli prikazana su na slici 22. i oni su jedini dostupni neregistriranim korisnicima. Za pristup preostalim modulima korisnik obavezno mora proći postupak registracije i prijave u sustav.

Slika 22. *Prikaz modula za registraciju i logiranje*



Na slici 23. je prikazan izgled modula „Kontakt“ koji služi za prikaz važnih informacija. Kao i na svim ostalim modulima mobilne aplikacije na vrhu se nalazi Bootstrap izbornik u obliku gumba. Nakon pritiska na gumb otvara se padajući izbornik koji sadrži kompletnu navigaciju, odnosno pristup svim modulima mobilne komponente. Kao i kod internet komponente, samo registrirani korisnici mogu pristupiti svim modulima putem navigacije, dok neregistrirani korisnici najprije moraju izraditi novi korisnički račun putem modula za registraciju da bi se mogli prijaviti u sustav.

Slika 23. *Modul za prikaz informacija i primjer Bootstrap izbornika*

9. SIGURNOST APLIKACIJE

9.1. Web.config datoteka

Konfiguracijski podaci kod ASP.NET web aplikacija pohranjeni su u obliku XML tekstualnih datoteka, a glavna datoteka se naziva Web.config [15]. Ona se nalazi u korijenskom direktoriju (engl. *root directory*) svakog projekta te sadrži specifične varijable i vrijednosti za tu aplikaciju. U sklopu Web.config datoteke zapisane su klijentske postavke, korisnička imena, lozinke i URL adrese servisa, te se vrlo lako mogu provjeriti, ažurirati ili dopuniti. Te vrijednosti zapisane su u XML datoteku pod appSettings element kao kolekcija stringova [16]. Kod web aplikacija im je moguće brzo pristupiti upotrebom klase WebConfigurationManager i naziva kolekcije iz Web.config datoteke. Vrijedi napomenuti da zato što su svi elementi tipa String neće doći do pogreške ukoliko prilikom čitanja neki podatak ne postoji u datoteci, već će se samo vratiti prazan string [17]. Primjer ispravnog unosa aplikacijskih postavki u Web.config datoteku s ciljem sigurnog dohvata unesenih vrijednosti prikazan je u kodu 9.

Glavne konfiguracijske postavke podijeljene su na više sekcija, a svaka od njih sadrži individualne postavke. Kako bi ASP.NET okvir uspješno kompajlirao MVC aplikaciju, potrebno je konfigurirati postavke u sekciji Compilation. Neki od atributa koje je moguće zadati su debug kojim određujemo želimo li da se generiraju simboli potrebni prilikom debugiranja (engl. *debugging*) i targetFramework za specifikaciju .NET okvira koji aplikacija koristi. Nakon postavljanja aplikacije na server poželjno je isključiti opciju debug u svrhu optimizacije performansi, dok targetFramework mora biti uključen samo kod aplikacija koje koriste .NET Framework 4 ili novije. ASP.NET okvir također može autorizirati podatke poput korisničkog imena i lozinke uz pomoć mode atributa koji se nalazi u sekciji za autorizaciju gdje se specificira autorizacijska shema aplikacije. ConnectionStrings sekcija pak služi za specificiranje kolekcije stringova koji služe za spajanje na bazu podataka. U postavkama je tako zapisan naziv stringa za povezivanje, server i naziv baze podataka, te korisničko ime i lozinka za pristup toj bazi.

Kod 9. Zapis aplikacijskih postavki u Web.config datoteci

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0"/>
    <add key="webpages:Enabled" value="false"/>
    <add key="ClientValidationEnabled" value="true"/>
    <add key="UnobtrusiveJavaScriptEnabled"
value="true"/>
    <!-- CLIENT SETTINGS -->
    <add key="ClientUri"
value="http://someapp.cloudapp.net/map/service.svc"/>
    <add key="ClientUsername" value="username"/>
    <add key="ClientPassword" value="password"/>
    <!-- EMAIL SETTINGS -->
    <add key="SenderEmail" value="mrisek@msn.com"/>
    <add key="SenderName" value="Admin"/>
    <add key="Server" value="mail.t-com.hr"/>
    <add key="EmailTitle" value="Kontrola unosa
dnevnica"/>
    <add key="EmailApproval" value="true"/>
    <!-- SMS SETTINGS -->
    <add key="SmsProviderLink"
value="http://www.kunderechner.net/public/smsgate.php"/>
    <add key="SmsUserName" value="1234"/>
    <add key="SmsPassword" value="password"/>
    <add key="SmsApproval" value="false"/>
  </appSettings>
...
</configuration>
```

9.2. Backup Azure SQL baze podataka

Microsoft Azure SQL Database sadrži ugrađeni backup i obnovu podataka kroz servise Point in Time Restore i Geo-Restore. Servis SQL Database automatski kreira backup svake aktivne baze podataka, što znači da se svakih sat vremena provede backup koji se tokom narednih 60 minuta može iskoristiti kroz Geo-Restore čime je omogućen RPO (engl. *Recovery point objective*) od jednog sata. Svrha Geo-Restore servisa je oporavak podataka u slučaju nedostupnosti baze zbog nemira ili većih incidenata u regiji gdje je smještena baza podataka. Point in Time Restore je pak moguć zahvaljujući backupu loga koji se vrši svakih 5 minuta. Riječ je o najosnovnoj opciji obnove podataka, a koristi se za oporavak baze na prethodno stanje od 7, 14 ili 35 dana unatrag, ovisno o pretplatničkom paketu što je i prikazano na slici 24. Dodatne opcije backupa su Database copy servis koji služi za kopiranje baze na isti ili neki drugi server neovisno o regiji, te Import and export servis kojim je moguće vlastoručno ili automatski unijeti i izvesti BACPAC datoteku pomoću koje se obnavlja baza podataka [19].

Slika 24. Opcije backupa u različitim pretplatničkim paketima Azure servisa [20]

Service Tier	Geo-Restore	Self-Service Point in Time Restore	Backup Retention Period	Restore a Deleted Database
Basic	Supported*	Supported*	7 days	Supported*
Standard	Supported*	Supported*	14 days	Supported*
Premium	Supported*	Supported*	35 days	Supported*
Web	Not supported	Not supported	n/a	n/a
Business	Not supported	Not supported	n/a	n/a

(https://msdn.microsoft.com/en-us/library/azure/jj650016.aspx)

9.3. Azure Scheduler Timeout Error

Kompletna logika provjere smještena je u zasebnoj klasi datoteke ProvjeraController.cs i tamo se nalazi akcijska metoda koja aktivira taj mehanizam na principu „fire and forget“ asinkrone operacije. Nova instanca pokreće se korištenjem Task.Factory.StartNew metode koja pozove samu provjeru i omogućava da Azure Scheduler uspješno pokrene skriptu za provjeru, a to je prikazano u kodu 10. Scheduler je inače ograničen na izvršavanje Job-ova koji traju manje od 30 sekundi, te se u slučaju prekoračenja tog limita cijeli postupak ponavlja. Tada dolazi do Scheduler Timeout-a te se u većini slučajeva zadatak ponavlja tako dugo dok se ne izvrši unutar ograničenog vremena ili se Job uopće ne izvrši zbog Scheduler Timeout Errora. Ta pogreška bila je najveći izazov za savladati tijekom razvoja aplikacije, a ideja za rješenje stigla je od strane Microsoft-ovog Supporta čija se sugestija pokazala ispravnom i učinkovitom.

Kod 10. Asinkrono pozivanje skripte za provjeru

```
namespace WebAplikacijaMvc.Controllers
{
    public class ProvjeraController : AsyncController{
        public ActionResult Provjera()
        {
            T.Task.Factory.StartNew(() =>
ProvjeraDnevnica.pokreniProvjeru());
            return View();
        }
        public class ProvjeraDnevnica
        {
            public static void pokreniProvjeru(){...}
        }
    }
}
```

10. ZAKLJUČAK

Ovim Završnim radom prikazana je integracija ASP.MVC aplikacije i nekoliko aktualnih servisa iz Microsoft Azure i Office 365 platforme koje su zajedno pridonjele pouzdanosti, suvremenosti i proširivosti samog sustava.Upotrebom navedenih tehnologija u cijelosti je postignut cilj izrade suvremene mobilne i web aplikacije koja služi za unos dnevnica od strane registriranih korisnika i sadrži implementiranu automatiziranu provjeru unosa istih. Samim time uspješno je napravljen moderni sustav koji služi za automatizaciju poslovnih procesa i dostupan je krajnjim korisnicima preko bilo kojeg uređaja.

Aplikaciju je moguće učiniti i kompleksnijom nego što je to prikazano u ovom radu. Za komercijalne potrebe istog projekta u njega su ugrađeni SMS servis i funkcionalnostmonitoringa tj. praćenje svake provedene akcije sustava. Dodana je i vlastita XML datoteka koja se redovito ažurira i služi kao log file. Pored toga, sustavu je dodan i administrator kojemu se nakon svakog pokretanja aplikacije na e-mail adresu šalju detaljni izvještaji monitoringa te vrijeme poslane elektroničke pošte i SMS poruka.

Zahvaljujući izradi ove suvremene aplikacije, potencijal korištenih tehnologija i platformi prepoznat je u tvrtci, napravljena je opsežna dokumentacija za njih te su se počele upotrebljavati i kod ostalih projekata.

11. LITERATURA

Pisani izvori:

- Galloway J. i sur. (2012.) Professional ASP.NET MVC 4, SAD, John Wiley & Sons Inc
- Chadwick J., Snyder T., Panda H. (2013.). Programiranje ASP.NET MVC4, Zagreb, Dobar Plan
- Esposito D. (2014.). Programming Microsoft ASP.NET MVC (3rd Edition), SAD, Microsoft Press
- Ciliberti J. (2013.). ASP.NET MVC 4 Recipes, SAD, Apress
- Freeman A. (2013.). Pro ASP.NET MVC 4, SAD, Apress

Internetski izvori:

[1] Microsoft WinDays portal – ASP.NET pregled novosti – Internet http://blog.windays.hr/asp-net-35151/(15.8.2015.)

[2] MVC Xerox Parc – Internet http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html (15.8.2015.)

[3] Microsoft Developer Network – ASP.NET MVC Overview – Internet https://msdn.microsoft.com/en-us/library/dd381412%28v=vs.108%29.aspx (15.8.2015.)

[4] Microsoft Developer Network – When to createMVC Application - Internet https://msdn.microsoft.com/en-us/library/dd381412%28v=vs.108%29.aspx (15.8.2015.)

[5] Mozilla Developer Network – JavaScript – Internet https://developer.mozilla.org/en-US/docs/Web/JavaScript (15.8.2015.)

[6] Mozilla Developer Network – Ajax - Internet https://developer.mozilla.org/en-US/docs/AJAX (15.8.2015.)

[7] jQuery – jQuery API – Internet http://api.jquery.com/ (15.8.2015.)

[8] Webopedia – Microsoft Office 365 – Internet

http://www.webopedia.com/TERM/O/office_365.html (15.8.2015.)

[9] Microsoft Office – What is Microsoft Exchange Service? – Internet https://support.office.com/en-au/article/What-is-a-Microsoft-Exchange-Server-account-36c6c5bb-e59a-4550-b320-04473482dc51?ui=en-US&rs=en-AU&ad=AU (15.8.2015.)

[10] Microsoft TechNet – Autodiscover Service – Internet https://technet.microsoft.com/en-us/library/bb124251%28v=exchg.150%29.aspx (15.8.2015.)

[11] Microsoft Azure – What is Azure - Internet https://azure.microsoft.com/en-us/overview/what-is-azure/ (15.8.2015.)

[12] Microsoft Azure – App services Overview – Internet https://azure.microsoft.com/en-us/documentation/articles/app-service-changes-existing-services/ (15.8.2015.)

[13] Microsoft Azure – SQL Database Overview – Internet https://azure.microsoft.com/en-us/documentation/articles/sql-database-technical-overview/ (15.8.2015.)

[14] Microsoft Azure – What is Azure Scheduler? – Internet https://azure.microsoft.com/en-us/documentation/articles/scheduler-intro/ (15.8.2015.)

[15] Microsoft Developer Network – ASP.NET Configuration Files – Internet https://msdn.microsoft.com/en-us/library/ms178684.aspx (15.8.2015.)

[16] MSDN – Format of ASP.NET Configuration Files – Internet https://msdn.microsoft.com/en-us/library/ackhksh7%28VS.71%29.aspx (15.8.2015.)

[17] Microsoft Developer Network – Configuration files – Internet https://msdn.microsoft.com/en-us/library/ackhksh7%28VS.71%29.aspx (15.8.2015.)

[18] MSDN – How to: Read Application Settings - Internet https://msdn.microsoft.com/en-us/library/610xe886.aspx (15.8.2015.)

[19] Microsoft Azure – Azure SQL Database Business Continuity – Internet https://msdn.microsoft.com/en-us/library/azure/hh852669.aspx (15.8.2015.)

[20] Microsoft Azure – Azure SQL Database Backup and Restore – Internet https://msdn.microsoft.com/en-us/library/azure/jj650016.aspx (15.8.2015.)